Project #1: RC5 Project

Mazen Muhammad Saad Ali Khodier (120180019)

March 25, 2021

Initialization of Parameters & Arbitrary Input

Initializing Memory Locations (8-bit Memory)

```
.EQU DATA_BYTES = W_PAIRS * 2 * WORD_BYTES            ; Size of User Data in Bytes
.EQU KEY_START = DATA_START + DATA_BYTES              ; Location of User's Key (After the Data)
.EQU EXPANDED_START = KEY_START + SECRET_BYTES        ; Memory Location of Expanded Key Array
.EQU ENCRYPTED_START = EXPANDED_START + TABLE_SIZE    ; Memory Location of Encrypted Data
.EQU DECRYPTED_START = ENCRYPTED_START + DATA_BYTES   ; Memory Location of Decrypted Data
```

Loading Arbitrary Data and Key into Memory

The user stores both the data they wish to encrypt/decrypt and the used key in the memory locations specfied above. ### Data

```
LDI ZL, LOW(DATA_START)
LDI ZH, HIGH(DATA_START)
```

Key

RC5 Algorithm Parameters (RC5-16/8/12 is Used)

```
.EQU ROUNDS = 8                 ; Number of Rounds (r)
.EQU TABLE_SIZE = 2*(ROUNDS+1)  ; Expanded Key Table Size (t)
.EQU WORD_BITS = 16             ; Word Size in Bits (w)
.EQU WORD_BYTES = WORD_BITS/8   ; Word Size in Bytes (u)
.EQU SECRET_BYTES = 12          ; Number of Bytes in the Secret Key (b)
.EQU SECRET_WORDS = 6           ; Number of Words in the Secret Key (c = Ceiling{b / u})
.EQU ITERATIONS = 54            ; Iterations in Key-Expansion Module (n = 3 * max{t, c})
.EQU PW = 0xb7e1                ; Key-Expansion Constant #1 ( Pw = Odd((e - 2) * 2^w) )
.EQU QW = 0x9e37                ; Key-Expansion Constant #2 ( Qw = Odd((phi - 1) * 2^w) )
```

Input-Related Parameters

```
.EQU DATA_START = 0x100         ; Arbitrary Memory Location of 1st byte of  User's Data
.EQU W_PAIRS = 3                ; Number of Word Pairs (A & B) User Wants to Encrypt/Decrypt
```

Initializing Memory Locations (8-bit Memory):

```
.EQU DATA_BYTES = W_PAIRS * 2 * WORD_BYTES                ; Size of User Data in Bytes
.EQU KEY_START = DATA_START + DATA_BYTES                  ; Location of User's Key (After the Data)
.EQU EXPANDED_START = KEY_START + SECRET_BYTES            ; Memory Location of Expanded Key Array
.EQU ENCRYPTED_START = EXPANDED_START + TABLE_SIZE * WORD_BYTES  ; Location of Encrypted Data
.EQU DECRYPTED_START = ENCRYPTED_START + DATA_BYTES       ; Memory Location of Decrypted Data
```

Some Useful Macros

```
.MACRO ADDW
ADD @0, @2
ADC @1, @3
.ENDMACRO

.MACRO GET_ROT
MOV @0, @1
ANDI @0, @2
.ENDMACRO

.MACRO ROLW
LOOP:
    BST @1, 7
    ROL @0
    ROL @1
    BLD @0, 0
    DEC @2
    BRNE LOOP
.ENDMACRO

.MACRO RORW
LOOP:
    BST @0, 0
    ROR @1
    ROR @0
    BLD @1, 7
    DEC @2
    BRNE LOOP
.ENDMACRO

.MACRO COUNTER
    LDI @0, @1
    LDI @2, LOW(@4)
    LDI @3, HIGH(@4)
.ENDMACRO

.MACRO XOR
EOR @0, @2
EOR @1, @3
.ENDMACRO

.MACRO SUBW
SUB @0, @2
SBC @1, @3
.ENDMACRO
```

Main Program

```
CALL LOAD_DATA
CALL LOAD_KEY
CALL EXPANSION
CALL ENCRYPTION
CALL DECRYPTION
JMP END
```

Loading Arbitrary Data and Key into Memory:

The user stores both the data they wish to encrypt/decrypt and the used key in the memory locations specfied above. Loading User's Data for Encryption/Decryption (3 Pairs of Words = 3*2 Words = 12 Bytes)

```
LOAD_DATA:
            LDI ZL, LOW(DATA_START)
            LDI ZH, HIGH(DATA_START)

            LDI R16, 0x4D
            ST Z+, R16
            LDI R16, 0x34
            ST Z+, R16
            LDI R16, 0x20
            ST Z+, R16
            LDI R16, 0x5A
            ST Z+, R16
            LDI R16, 0x61
            ST Z+, R16
            LDI R16, 0x79
            ST Z+, R16
            LDI R16, 0x20
            ST Z+, R16
            LDI R16, 0x4D
            ST Z+, R16
            LDI R16, 0x61
            ST Z+, R16
            LDI R16, 0x7A
            ST Z+, R16
            LDI R16, 0x65
            ST Z+, R16
            LDI R16, 0x6E
            ST Z+, R16
            RET
```

Loading User's Key for Expansion (12 Byte-Long Key)

```
LOAD_KEY:
            LDI ZL, LOW(KEY_START)
            LDI ZH, HIGH(KEY_START)

            LDI R16, 0x2E
            ST Z+, R16
            LDI R16, 0x20
            ST Z+, R16
            LDI R16, 0x4D
            ST Z+, R16
            LDI R16, 0x61
            ST Z+, R16
            LDI R16, 0x7A
            ST Z+, R16
            LDI R16, 0x65
            ST Z+, R16
            LDI R16, 0x6E
            ST Z+, R16
            LDI R16, 0x20
            ST Z+, R16
            LDI R16, 0x41
            ST Z+, R16
            LDI R16, 0x6B
            ST Z+, R16
            LDI R16, 0x74
            ST Z+, R16
            LDI R16, 0x72
            ST Z+, R16
            RET
```

Key Expansion Sub-Routine

```
EXPANSION:
            ; First Step isn't necessary as data is accessed byte-wise onl
            ; Second Step
            LDI ZL, LOW(EXPANDED_START)
            LDI ZH, HIGH(EXPANDED_START)
            LDI R16, LOW(PW)
            LDI R17, HIGH(PW)
            ST Z+, R16
            ST Z+, R17
            LDI R18, LOW(QW)
            LDI R19, HIGH(QW)
            LDI R20, TABLE_SIZE - 1
            SECOND:
                ADDW R16, R17, R18, R19
                ST Z+, R16
                ST Z+, R17
                DEC R20
                BRNE SECOND

            ; Third Step
            LDI R17, SECRET_WORDS           ; j counter
            LDI ZL, LOW(KEY_START)          ; j
            LDI ZH, HIGH(KEY_START)
            LDI R16, TABLE_SIZE             ; i counter
            LDI YL, LOW(EXPANDED_START)     ; i
            LDI YH, HIGH(EXPANDED_START)

            LDI R18, ITERATIONS
            LDI R19, 0  ; AL
            LDI R20, 0  ; AH
            LDI R21, 0  ; BL
            LDI R22, 0  ; BH
            LDI R23, 0  ; SiL
            LDI R24, 0  ; SiH
            LDI R25, 0  ; LjL
            LDI R26, 0  ; LjH
            LDI R27, 0  ; Shift Amount
            THIRD:
                ADDW R19, R20, R21, R22
                LDI R27, 3
                LD R23, Y+
                LD R24, Y+
                ADDW R19, R20, R23, R24
                ROLW R19, R20, R27
                ST -Y, R20
                ST -Y, R19
                LD R24, Y+
                LD R24, Y+

                ADDW R21, R22, R19, R20
                GET_ROT R27, R21, 0x0F
                LD R25, Z+
                LD R26, Z+
                ADDW R21, R22, R25, R26
                TST R27
```

```
    BREQ SKIP1
    ROLW R21, R22, R27
    SKIP1: ST -Z, R22
    ST -Z, R21
    LD R26, Z+
    LD R26, Z+

    DEC R16
    BRNE SKIP2      ; Update pointers in case r16 = 0
    COUNTER R16, TABLE_SIZE, YL, YH, EXPANDED_START
    SKIP2: DEC R17
    BRNE SKIP3      ; Update pointers in case r17 = 0
    COUNTER R17, SECRET_WORDS, ZL, ZH, KEY_START

    SKIP3: DEC R18
    BRNE THIRD
RET
```

Encryption Sub-Routine

```
ENCRYPTION:
            LDI YL, LOW(DATA_START)
            LDI YH, HIGH(DATA_START)
            LDI ZL, LOW(ENCRYPTED_START)
            LDI ZH, HIGH(ENCRYPTED_START)
            LDI R24, W_PAIRS
            PAIR:
                LD R16, Y+  ; AL
                LD R17, Y+  ; AH
                LD R18, Y+  ; BL
                LD R19, Y+  ; BH
                LDI XL, LOW(EXPANDED_START)             ; iL
                LDI XH, HIGH(EXPANDED_START)            ; iH
                LD R21, X+
                LD R22, X+
                ADDW R16, R17, R21, R22
                LD R21, X+
                LD R22, X+
                ADDW R18, R19, R21, R22
                LDI R23, ROUNDS
                ENC:
                    XOR R16, R17, R18, R19
                    GET_ROT R20, R18, 0x0F
                    TST R20
                    BREQ SKIP4
                    ROLW R16, R17, R20
                    SKIP4: LD R21, X+
                    LD R22, X+
                    ADDW R16, R17, R21, R22


                    XOR R18, R19, R16, R17
                    GET_ROT R20, R16, 0x0F
                    TST R20
                    BREQ SKIP5
                    ROLW R18, R19, R20
                    SKIP5: LD R21, X+
                    LD R22, X+
                    ADDW R18, R19, R21, R22
                    DEC R23
                    BRNE ENC
                ST Z+, R16
                ST Z+, R17
                ST Z+, R18
                ST Z+, R19
                DEC R24
                BRNE PAIR
            RET
```

Decryption Sub-Routine

DECRYPTION: LDI ZL, LOW(DECRYPTED_START)
LDI ZH, HIGH(DECRYPTED_START)
LDI YL, LOW(ENCRYPTED_START)
LDI YH, HIGH(ENCRYPTED_START)
LDI R24, W_PAIRS
INPUT: LD R16, Y+ ; AL LD R17, Y+ ; AH LD R18, Y+ ; BL LD R19, Y+ ; BH

```
            LDI XL, LOW(ENCRYPTED_START)              ; iL
            LDI XH, HIGH(ENCRYPTED_START)             ; iH
            LDI R25, ROUNDS
            DECR:
                LD R21, -X  ; iH
                LD R20, -X  ; iL
                SUBW R18, R19, R20, R21
                GET_ROT R23, R16, 0x0F
                TST R23
                BREQ SKIP6
                RORW R18, R19, R23
                SKIP6: XOR R18, R19, R16, R17

                LD R21, -X  ; iH
                LD R20, -X  ; iL
                SUBW R16, R17, R20, R21
                GET_ROT R23, R18, 0x0F
                TST R23
                BREQ SKIP7
                RORW R16, R17, R23
                SKIP7: XOR R16, R17, R18, R19

                DEC R25
                BRNE DECR
            LD R23, -X  ; 1H
            LD R22, -X  ; 1L
            LD R21, -X  ; 0H
            LD R20, -X  ; 0L
            SUBW R18, R19, R22, R23
            SUBW R16, R17, R20, R21
            ST Z+, R16
            ST Z+, R17
            ST Z+, R18
            ST Z+, R19

            DEC R24
            BRNE INPUT
        RET
```

END: NOP